The 2024 ICPC Asia EC Regionals Online Contest (II) Tutorial

杭州电子科技大学命题组

2024 年 9 月 21 日

A. Gambling on Choosing Regionals

注意到最坏情况就是你去哪里强队都去哪里，因此越小的赛站可能的排名就越低，只需要关注最小的站。

按照能力值排序，从大到小扫描即可，每个学校最多保留 $\min c_i$ 个队伍。时间复杂度 $O(n \log n)$。

B. Mountain Booking

从 1 到 m 依次考虑每个日期。假设当前正在考虑第 i 天，那么只有第 i 天来访的游客以及指定第 i 天的查询是有用的。将这些游客和查询都提取出来，通过 Kruskal 重构树可以很方便地在 $O(n \log n)$ 的时间内计算出这些查询的答案。

不幸的是，本题还有加边删边操作，无法轻易地动态维护 Kruskal 重构树。解决问题的关键是注意到假设第 i 天有 t_i 个游客、q_i 个询问，那么可以支付 $O((t_i + q_i) \log n)$ 的代价来获取它们对应的节点形成的大小为 $O(t_i + q_i)$ 的虚树，然后在虚树上暴力构建 Kruskal 重构树计算每个询问的答案。

求虚树的方法很多，比如 LCT 或者离线分治。假设 n, m, p, q 同阶，总时间复杂度为 $O(n \log n)$。

C. Prefix of Suffixes

如果使用 exkmp 是只能求一个静态字符串的 z 数组，下面简述了如何用 kmp 实现一个动态的 exkmp。

kmp 可以做到动态向后添加元素并且能求出每个前缀的最长 border，记 $border_i$ 为 $s[1 \dots i]$ 的最长 border，下面考虑如何使用 border 数组来求出 z 数组。

令 z_i 表示 i 开始的位置和前缀的最长公共前缀长度，那么 z_i 的情况有两种：

第一种情况是 z_i 已经定下来了（并且之后不可能再被修改）。

第二种情况是还没有定下来，向后添加一个字符会使得 z_i 长度增加 1 / 定下来。

根据上面的分析，只需要知道每次向后添加一个字符会定下来哪些 z_i 的值即可，并且每个值最多被确定一次。

如果我们添加了一个 $s[n] = c$，对于没有被定下来的 z_i ，如果 $s[n-i+1] \neq s[n]$ 就可以将 z_i 定为 $n - i$。

目标是如何快速的找到所有这样的 i，同时满足条件 $(s[1 \dots n-i] = s[i \dots n-1])$ 和 $(s[n-i+1] \neq s[n])$。

假设将 $i \to border_i$ 连边成一棵树，那么可以知道满足第一个条件的 $n-i$ 一定是 $n-1$ 在树上的祖先节点。

对第二个条件可以做一个简单的处理，例如用一个 $fa_{x,j}$ 表示 x 第一个满足条件 $s[y+1]=j$ 的祖先节点 y，那么只需要对于 $j \neq c$，从 $n-1$ 这个点暴力跳 $fa_{x,j}$ 即可，就可以得到同时满足两个条件的 $n-i$ 的值。

由于每个点的 z_i 只会被确定一次，所以暴力跳的复杂度是正确的。

剩下的部分是容易的，只需要记录每个时刻 i 没有被定下来的 z_j 对应的 b_j 的和，和 a_i 简单相乘即可。

$O(n\log n)$ 可以通过，精细一点的实现可以做到 $O(n)$。

D. Query on Tree

我们维护三个数组 a_x,b_x,t_x，a_x 是点 x 的点权，b_x 是点 x 的所有 10 级子孙的点权最大值，t_x 是点 x 的所有 10 级子孙的点权增量懒标记。

对于第一种操作和第二种操作，我们从点 x 处往上爬 $k-1$ 级祖先，每一级祖先涉及到的点都是在 bfs 序上连续的一段或两段区间，总共就是 $O(k)$ 段区间。

对于 a_x 按照 bfs 序排列并用线段树维护即可。

而对于 b_x 和 t_x，注意到我们操作的每一段 bfs 区间内的所有点的 10 级祖先都是相同的，只需要修改或查询 $O(k)$ 个点对应的 b_x 和 t_x 就可以了。

对于第三种操作，我们将点 x 的子树内的点分为两类，一类是距离点 x 小于 10 的，一类是距离点 x 不小于 10 的。

对于第一类，在 bfs 序上仍然是 10 段区间，且每一段的所有点的 10 级祖先仍然相同，类似第一、二种操作维护即可。

对于第二类，所有点的 10 级祖先在 dfs 序上是一个区间，直接按照 dfs 序排列 b_x,t_x 并用线段树维护即可。

单组数据时间复杂度 $O(nk+qk\log n)$。

E. Escape

可以看成 Sneaker 和杀戮机器人都不能在原地停留，然后杀戮机器人有个活动范围限制。

如果 Sneaker 和杀戮机器人可以在原地停留，那么 Sneaker 到达一个点肯定会尽可能早，而且时间必须比杀戮机器人到达这个点短。

那么预处理一下每个点最早什么时候会被杀戮机器人到达，然后在这个基础上处理出 $1\sim n$ 的最短路即可。

由于所有杀戮机器人的活动半径是相同的，我们可以用多源最短路完成预处理。

现在考虑 Sneaker 和杀戮机器人都不能在原地停留。此时我们发现通过在相邻边上反复横跳，可以让杀戮机器人达到一个点的时间加 2，那么我们处理达到一个点奇数时间的最小值和偶数时间的最小值，就可以套上面的做法了。

单组数据时间复杂度 $O(n+m)$。

F. Tourist

$O(n)$ 模拟即可，注意多个负数的累积也有可能导致溢出。

G. Game

平局其实是没有用的，可以认为双方的胜利概率就是 $a_0 : a_1$，即 $p_0 = \frac{a_0}{a_0+a_1}, p_1 = \frac{a_1}{a_0+a_1}$。

双方的游戏过程实际上是一个辗转相减的过程，用辗转相除加速即可。

时间复杂度 $O(T\log(n+m))$。

H. Points Selection

注意到如果 $query(a,b,c)$ 为真，那么 $query(\geq a, \geq b, c)$ 一定为真。

从小到大枚举询问中 a 的值，按横坐标从小到大依次加入每个点，维护 f_c 表示最小的 b 满足 $query(a,b,c)$ 为真。假设当前正在加入点 (x,y,w)，有 $f_{(c+w) \bmod n} = \min(f_{(c+w) \bmod n}, \max(f_c, y))$。

观察状态转移方程可知，如果 $y \geq f_{(c+w) \bmod n}$，那么就没有更新的必要。令 $m = \max(f_0, f_1, \ldots, f_{n-1})$，那么如果 $y \geq m$，则这个点是完全无用的，可以直接跳过。

由于数据随机，可以近似地认为加入 k 个点后，所有 2^k 个子集和模 n 的结果在 $[0,n)$ 等概率均匀分布。可以看作 2^k 个小球随机放入 n 个洞之中，填满所有 n 个洞所需的期望球数是 $O(n\log n)$，因此 $k = O(\log n)$ 时期望可以填满整个 f 数组。这说明，f 数组的最大值的期望值等于所有已经加入的点的纵坐标的第 $O(\log n)$ 小值，即 $O(\frac{n\log n}{k})$。

于是，加入的第 k 个点的纵坐标 $y < m$ 的概率为 $O(\frac{\log n}{k})$，期望只需要更新 $O(\sum_{k=1}^{n} \frac{\log n}{k}) = O(\log^2 n)$ 遍 f 数组。由于这个值并不大，每次暴力 $O(n)$ 更新整个数组即可。有了 f 数组就可以很容易地求出最终的答案。

时间复杂度 $O(n\log^2 n)$。

I. Strange Binary

首先，我们可以证明 4 的倍数一定不可以分解，因为此时必然有 $a_0 = a_1 = 0$。

否则，可以先将序列 a 全赋值为 1，然后再根据 $2^{32} - 1 - x$ 的在二进制下 2^i 那一位决定 a_{i-1} 是否变为 -1。

此时需要修正的部分是 x 为偶数的情况，此时将 a_0 变为 0 就可以了。

时间复杂度 $O(T\log n)$。

J. Stacking of Goods

微扰法证明最优顺序一定是按照 $\frac{c_i}{w_i}$ 不增的顺序摞起来的。排序即可。复杂度 $O(n\log n)$。

K. Match

从高位到低位依次枚举。

$f(A,B,k)$ 表示 A 集合和 B 集合进行 $A_i \oplus B_j \geq k$ 匹配的答案，返回一个多项式，依次表示 $0,1,2,3,\ldots,\min(|A|,|B|)$ 个匹配的方案数。

假设 k 这一位的权值为 0，那么所有 A_i, B_j 只要当前位不同就可以进行匹配。

我们把 A 按当前位为 0 或 1 分类成 A_0, A_1，B 同理分成 B_0, B_1，先算出 $f(A_0,B_0,k), f(A_1,B_1,k)$，剩下的匹配在 A_0, B_1 和 B_1, A_0 中都可以任意匹配，可以组合数 DP 转移。

假设 k 这一位的权值为 1，那么只有 $f(A_1,B_0,k)$ 和 $f(A_0,B_1,k)$ 有解，将二者最高位抹去继续递归 DP 即可。

时间复杂度为 $O(n^4)$，常数很小。

L. 502 Bad Gateway

首先可以将问题看成：每一步可以选择重置成 $[0,t)$ 中给某个整数，或者减 1，求最优策略下的得到 0 的期望步数。

那么最优的策略一定是选择一个阈值 $c \in [1,t]$，开始不停重置，一旦重置到一个小于 c 的数就直接减到 0。

设重置到小于 c 的数的概率为 $p=\frac{c}{t}$，根据期望线性性拆解到计算重置 k 次的概率 $(1-p)^k$ 之和，那么这个策略下的期望步数为：

$$\frac{c-1}{2}+1+(1-p)+(1-p)^2+\cdots=\frac{c-1}{2}+\frac{t}{c}$$

容易发现这是一个对勾函数的形式，最小值取在 $c=\left\lfloor\sqrt{2t}\right\rfloor$ 或 $c=\left\lceil\sqrt{2t}\right\rceil$ 处。

实现时需要分数类，考虑到分数类中的最大公约数计算，时间复杂度 $O(T\log n)$。